Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5591--07-10,156

Session Initiation Protocol Network Encryption Device Plain Text Domain Discovery Service

Christopher L. Robson

Center for Computational Science

Information Technology Division

December 7, 2007

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
07-12-2007

2. REPORT TYPE
Formal Report

3. DATES COVERED *(From - To)*
August 30 to November 30, 2007

4. TITLE AND SUBTITLE
Session Initiation Protocol Network Encryption Device Plain Text Domain Discovery Service

5a. CONTRACT NUMBER
55-9118-B-7

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0603648D8Z

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

6. AUTHOR(S)
Christopher L. Robson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL/FR/5591--07-10,156

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Steve Sligar, JCTD Program Manager
National Geospatial-Intelligence Agency, Reston
12310 Sunrise Valley Drive, Reston, VA 20191-3449

10. SPONSOR / MONITOR'S ACRONYM(S)
NGA

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Record and Disclosure of Invention Part II Disclosure of Invention

14. ABSTRACT

This report provides a method for cryptographic isolated domains to discover other cryptographic isolated domains by using the IETF Session Initiation Protocol (SIP). This method, called the SIP Network Encryption Device Plain Text Domain Discovery Service (SIP-DS), will not require a new IETF standard or any modification to existing IETF standards, nor are any specifically configured infrastructure or network devices required. This discovery method allows any encryption device, be it typical U.S. government Type 1 encryption devices such as TACLANE, HAIPE, or FASTLANE, or any non-government cryptographic devices implementing this technology, to find and exchange plain text domain (PTD) information. Additionally, SIP-DS will allow one encryption device to proxy PTD information for other encryption devices unable to implement this method.

15. SUBJECT TERMS
Session Initiation Protocol network encryption device plain text domain discovery, exchange, and proxy information service

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
Unclassified	Unclassified	Unclassified	UL	29	Christopher Robson

19b. TELEPHONE NUMBER *(include area code)*
(202) 404-3138

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

CONTENTS

SESSION INITIATION PROTOCOL NETWORK ENCRYPTION DEVICE PLAIN TEXT DOMAIN DISCOVERY SERVICE

1. INTRODUCTION

Several proposals have been suggested on how to solve the High Assurance Internet Protocol Encryptor (HAIPE) discovery problem. Existing encryption devices, such as the TACLANE, have developed unique discovery methods to establish Plain Text Domain (PTD) Security Associations (SA). All of these techniques require some change to either existing standards such as Internet Engineering Task Force (IETF) routing standards or network service standards as in the case of using a modified Domain Name Service (DNS). All of these suggestions result in government uniquely tailored protocols. For example, one proposal for the HAIPE problem suggests using a modified version of Border Gateway Protocol (BGP), while another promotes using a special DNS configuration. In the case of TACLANEs, using a BGP design would require a redesign of the TACLANE IP protocol support. Further, DNS presents several challenges if the PTD supports more than one network/mask or host/mask address. What is needed is a common solution that does not require making changes to existing standards or uniquely network configured architectures and provides a common technology solution for all encryption devices.

2. PURPOSE

This report suggests a new approach to solving the PTD SA discovery problem using an existing IETF standard "as is." This report proposes a mechanism for any encryption device, be it TACLANE or HAIPE, to find and establish SAs for associated PTDs.

3. SCOPE

This report focuses on a conceptual idea, presents a design, provides a concept of operation example of how the architecture will function, and finally proposes a prototype implementation.

4. WHAT IS THE SESSION INITIATION PROTOCOL?

The Session Initiation Protocol[1] (SIP) is a signaling protocol used to determine a participant's location, availability and capabilities, and assist in the setup process and session management. SIP, itself, doesn't provide services but provides several methods (or primitives) for clients or servers to find and use services. One common method a SIP participant uses to advertise to other participants is by registering to the SIP Registrar so that each client's information can be distributed within the SIP domain and end-to-end calls can be established. A common SIP element is the proxy server. Typically, the SIP proxy will route SIP connection requests and traffic between registered participants. The proxy also reports a member's location, authenticates and authorizes the member for services, implements provider call-

[1] It is not within the scope of this specification to detail or even provide a brief overview of the SIP protocol, how it works, or what are the SIP components. It is assumed the reader is familiar with SIP or will have done the necessary background SIP research (read at least Refs. 1, 2, and 3) to understand how this architecture is exploiting SIP.

Manuscript approved August 30, 2007.

routing policies, and provides features to SIP participants. Another method used for SIP call establishment is to send a SIP INVITE message to a peer system. The INVITE is used as a function of a call setup and used sometimes in a peer-to-peer (P2P) session establishment or via the SIP proxy. SIP is a request/response protocol (designed similarly to HTTP) and contains a set of response codes. These responses include the ACK, CANCEL, BYE, and options messages. Additionally, SIP has a suite of interoperable services collectively known as instant messaging and presence (IMP) described in Request for Comment (RFC) 2779 and in the Common Presence and Instant Messaging (CPIM) specification. These services are commonly referred to as SIP for Instant Messaging and Presence Leveraging Extension (SIMPLE). It is SIMPLE that this proposed architecture will exploit and that this report describes.

5. WHY SIP AS A PTD DISCOVERY PROTOCOL?

SIP is a signaling protocol used for establishing sessions in an IP network. SIP does not know about the details of a session nor is it in the path of the data; it just initiates, terminates, and modifies sessions. This simplicity means that SIP scales, is flexible, and sits comfortably in different architectures and deployment scenarios such as providing a vehicle for encryption devices to discover each other while not interfering with, being part of, or adding complexity to, the encryption function. This design focuses on several key SIP characteristics:

a. Reusing Existing Protocols — SIP is designed to specifically reuse as many existing protocols and protocol design concepts as possible. For example, SIP is modeled after Hypertext Transfer Protocol (HTTP), using Uniform Resource Locators (URL) for addressing and Session Description Protocol (SDP) to convey session information. SDP can be easily exploited to support various encryptor discovery requirements yet not require any SIP or associated protocols to be modified.
b. Maximizing Interoperability — SIP is also designed so that it would be easy to bind SIP functions to existing protocols and applications, such as e-mail and Web browsers. SIP does this by limiting itself to a modular philosophy — just like many other Internet protocols — and focusing on a specific set of functions. This will allow any encryptor architecture to use this protocol to exchange state and discovery peering points.
c. SIP REGISTER, INVITE, and event notification messages, which include MESSAGE, PUBLISH, SUBSCRIBE, and NOTIFY, are ideal for conveying Plain Text Domain information. Each has existing message fields that can be used to exchange PTD information. For example, the event notification "Presence Event Package" message can contain Presence Information Data Format (PIDF) documents containing detailed PTD information, which can include network and host Internet Protocol (IP) addresses, Information System Security Office (ISSO) point of contact information and PTD status information. The PIDF could provide Security Association key data. Further, SIP Common Profile for Presence (CPP) semantics and data formats can be exploited to facilitate the exchange of PIDF contained PTD information.

SIP was designed to work with a broad spectrum of existing and future IP protocols. To this end, SIP provides four basic functions. SIP allows for the advertising of user location, in this architecture this translates to the network address of the encryption device. Therefore, SIP provides a way for encryption devices to determine peers. SIP provides for feature negotiation so that all of the participants in a session can agree on the features to be supported among them. This same mechanism can be used to determine the networks or hosts the encryption device protects within its PTD. Thus the PTD information can be exchanged between Plain Text Domains via SIP. Therefore, the encryption device can establish Security Associations. SIP is a mechanism for call management — for example adding, dropping, or transferring participants — so the encryption device will not only be able to use existing standards as a solution for

peering but can extend this standard on how future SAs will be negotiated or renegotiated. Finally, SIP allows for changing features of a session while it is in progress. This ability to renegotiate sessions might be exploited for key management such as SA rekeying.

6. PTD ENCYPTION DEVICE DISCOVERY DESIGN REQUIREMENTS

Today's encryption device specifications are good examples of encryption device requirements for reachability and discovery. Generally, these requirements are common for almost all network encryptors. These requirements typically specify the following:

a. The protocol shall have a Registration function for encryption devices to advertise local Reachability Information.
b. The protocol shall have a Solicitation function for encryption devices to query and respond for remote prefix Reachability Information.
c. The protocol shall be bandwidth efficient.
d. Encryption devices shall send registration messages upon startup and every time a change is detected in the Local Enclave Prefix Table.
e. The encryption device shall include a lifetime associated with the Prefixes it registers.

7. HOW SIP UNIFORM RESOURCE IDENTIFIER (URI) WOULD BE USED FOR PTD ENCRYPTION DEVICE DISCOVERY

A SIP URI identifies a communications resource. In the case of Encryptor Discovery, the URI would be used to identify a unique encryption device with the SIP IA services database, thus providing the information for advertising reachability information. The URI contains sufficient information to initiate and maintain a communication session within the encryptor discovery process.

8. OTHER BENEFITS OF SIP PROTOCOL MESSAGES

Because SIP has well-defined messages, these can be exploited transparently without change for encryption device discovery and communications control services. SIP messages can be used by the encryptor for request or response messages in the PTD discovery process. Because there is no change to the existing SIP standard, existing client and server software can be reused. For example, SIP message types and options would take on the following meanings without modifying the existing SIP protocol.

a. Request Methods

   i. **INVITE** can be used to initiate an encryption device pair discovery and could also be used to change SA parameters once a session has been established. This can be thought of as a re-INVITE. The INVITE message coupled with Session Description Protocol will provide the method to update the User Agent Server (UAS) with Plain Text Domain Security Associations.
   ii. **SUBSCRIBE** is a request from a User Agent Client (UAC) (Presence User Agent or PUA) to establish a SA with another UAC through the UAS (Presence Agent or PA).
   iii. **MESSAGE** is used to carry PTD information message from the UAC (PUA) to a peering PUA.
   iv. **PUBLISH** is used to carry PTD information message from the UAC (PUA) to a UAS (Presence Server or PS) or directly to a P2P UAC.
   v. **NOTIFY** informs the UAC (PUA) that there is SA information available for a subscriber.
   vi. **ACK** would be an acknowledgment and confirmation response to an encryption device discovery invite.

vii. **BYE** would terminate a SA in progress.
viii. **CANCEL** would terminate the encryption device discovery invite and stops all searches and devices that are in the discovery or SA establishment process.
ix. **OPTIONS** would be used to query the capabilities of an encryption device.
x. **REGISTER** would register an encryption device's location with a server.
xi. **INFO** could be used to send information during a session without modifying the session state.

b. Response Message Types

xii. **1xx** provisional, searching, ringing, or queuing an encryption device discovery
xiii. **2xx** successfully discover an encryption device
xiv. **3xx** redirection, forwarding of an encryption device discovery invite
xv. **4xx** request failure (client)
xvi. **5xx** server failure
xvii. **6xx** global failure (busy, refusal, not available anywhere)

9. HOW SIP REGISTRATION WOULD BE USED FOR PTD ENCRYPTION DEVICE REGISTRATION

The encryptor, by definition, hides networks and systems within the encryptor's PTD; therefore, a caller doesn't know the address of the callee's fronted encryption device. That is, it doesn't know the encryption device-to-PTD binding. The SIP proxy servers for the PTD, also referred to as User Agent Servers or UAS, would perform the job of finding out the exact location of the end recipient and the associated encryptor. What actually would happen is that each encryption device, like a typical SIP client, would register its current location to a SIP REGISTRAR server. The encryptor will send a message called REGISTER, like a typical SIP client, informing the server of its present location. The Registrar stores this binding (between the encryption device and its present address) in a location server that is then used by other proxies or encryptors to locate the targeted encryptor that is servicing a particular PTD.

10. HOW SIP REGISTER WOULD CONVEY PTD

The encryptor can exploit the SIP REGISTER message Path header field, defined in Request for Comment (RFC) 3327, to establish the initial PTD database. An advantage to using the REGISTER Path header field is quick updating of the UAS PTD database from the onset of an encryption device registration process. As defined in RFC 3327, the Path header field structure will not change in format. That is, the field will still be represented as <sip:Uniform Resource Identifier (URI)>. However, the Path header field will now represent an ordered list of route elements that are used to build a Contact field route list. It will be used to represent a list of PTD addresses with the URI field elements in the form: URI = <PTD@host encryptor>. Within this context, the PTD field is defined as: PTD = <IP address/network mask>. For example, for the PTD with network IP address 192.168.1.0 that is protected by an encryptor with the IP address of 10.1.1.1, the REGISTER Path header field would be represented as "sip:192.168.1.0/24@10.1.1.1." As with any REGISTER message containing Path header fields, multiple Path header entries may be defined. For example, the REGISTER message with a Path header element would be as follows:

REGISTER sip:FQDN SIP/2.0
Via: SIP/2.0/UDP [IP address]:5060;branch=[information]
To: User Agent <sip:URI>
From: User Agent <sip:URI>;tag=[number]

Call-ID: URI
CSeq: [number] REGISTER
Contact: <sip:URI>
Supported: path
Path: <sip:URI>
. . .

Because the Path header function is to convey PTD information between the UAC and the UAS to which it is bound, the UAC REGISTER message does not traverse through any SIP proxies. Otherwise, traversed proxies may assume, correctly, that the message is a typical REGISTER message and attempt to modify the Path header field. If any proxy modifies the SIP Path header, this new, non-PTD information would be processed by the target UAS into PTD database information, thus corrupting the PTD database with incorrect information. Because using the Path header field does introduce some complexity in the PTD database update process, it is assumed this method will not be used as the default method for updating the UAS PTD database.

However, the use of a REGISTER message with a Path header might be used by a tactical UAC that is at the end of a limited bandwidth satellite communication link in order to reduce SIP message traffic overhead. This method would reduce the tactical UAC PTD information database initialization by at least four strings of variable data overhead required within the preferred IETF presence event package PIDF method discussed in Section 13.

11. HOW SIP INVITE SESSION DESCRIPTION PROTOCOL WOULD CONVEY PTD

The encryptor will exploit the SIP invite message, INVITE, to establish an information database from which clients can request information about peering networks and establish SA communication sessions between PTD clients or networks. The UAC will update the UAS with its PTD reachability information using the INVITE SDP. In this way, the encryptor does not have to maintain a database of client-to-client or even client-to-server table entries. However, where the UAC is not registering with a UAS, the client will maintain a P2P database similar to the UAS, therefore establishing reachability using SIP P2P. Using the same UAC to UAS dialogue defined within the SIP SDP standard, the encryptor will direct a PTD connection request to the SIP server (or UAC in the case of P2P SIP) using the INVITE SDP dialogue exchange. The SDP format of <type> = <value> will be maintained within the information data. Specifically, the IETF RFC 2327 field “c=” will be used to convey the encryptor PTD information. The SDP fields used will contain the following information:

V = protocol version
O = session identifier
S = session name
I = session information
U = URI in the form name@network address.
E = ISSO email address
P = ISSO phone number
C = PTD network/mask and/or system/mask
. . .

12. HOW SIP INVITE WOULD BE USED TO OBTAIN PTD INFORMATION

The UAS will use the requesting encryptor's (UAC) SIP INVITE message "To" and "Contact" information fields to extract UAS PTD database entries. This INVITE message will not contain any SDP information because it is not to be used as a PTD initialization or update message.

13. HOW SIP PRESENCE EVENT PACKAGE (PEP) PRESENCE INFORMATION DATA FORMAT (PIDF) WOULD CONVEY PTD

Presence Event Package PIDF documents will provide the most flexible, verbose information content SIP messages for use in encryption device discovery. Therefore, it is the preferred method for exchanging PTD information. The PTD PEP PIDF document will adhere to the World Wide Web Consortium (W3C) EXtensible Markup Language (XML) standard, and the message packet format will adhere to the IETF standards RFC 3265 and RFC 3856. As the IETF standard defines, the subscribing or subscribed encryptor(s) will act as the PUA and the SIP server(s) would function as the Presence Server (PS) or Agent (PA). The PUA and PS will use SIP MESSAGE, PUBLISH, SUBSCRIBE, and NOTIFY messages to convey PTD information, acknowledgments, and presences.

13.1 MESSAGE PTD Database Exchange

The PUA will use a SIP MESSAGE to convey to a peering PUA its PTD information when authentication is not required or a PS is not available. SIP MESSAGE communications functions similarly to User Datagram Protocol (UDP) connectionless information exchange. A key benefit to using this SIP method is in low bandwidth situations such as tactical radio environments where reduced data and control plane traffic are required. Figure 1 illustrates the information contained in the MESSAGE information packet (note: "value" represents a variable runtime field). As with any SIP PEP PIDF message, the PTD information is contained within the MESSAGE "Message Body" field.

13.2 PUBLISH PTD Database Exchange

The PUA will use a PUBLISH message to convey to a PS its PTD information. Figure 2 illustrates the information contained in the PUBLISH message (note: "value" represents a variable runtime field). As with any SIP PEP PIDF message, the PTD information is contained within the PUBLISH "Message Body" field.

13.3 SUBSCRIBE PTD Database Requests

The SUBSCRIBE message is used by an encryptor to discover from the PS the association between the PTD end system and its encryptor. The "Request-Line" field contains the URI of the target end system in the form **[end system IP address]@[PTD]**. This URI would consist of the target end system IP address prefixed with the PTD SIP domain. For example, if a subscribing encryptor received traffic from one of its PTD hosts with the IP of "10.1.1.1" and the PS SIP domain is "PTDSIPDOMAIN," the Request-Line URI would read "10.1.1.1@PTDSIPDOMAIN." Used slightly differently than a typical SIP SIMPLE message exchange, the "To" field will contain the IP address of the PS. This is because the SUBCRIBE message is really addressed to the PS, asking the PS for PTD information. The SUBSCRIBE message may or may not contain a PEP PIDF document. If the PIDF is present in the SUBSCRIBE message, it is used to convey additional information pertinent to the SUBSCRIBE message. Therefore, the PIDF "encryptor" value field is set to "0.0.0.0" and the "PTD_IP_address" field must be set to the end system's IP address. The format and content of the SUBSCRIBE message are described in Fig. 3.

```
Request-Line: MESSAGE sip:[subscribing PUA IP address]@[PTD] SIP/2.0
    Method: MESSAGE
    [Resent Packet: False]
Message Header
    Via: SIP/2.0/TCP [PS IP address]:5060;branch=[value]
    From: <sip:[subscribing PUA]@[PTD]>;tag=[value];epid=[value]
    To: <sip:[PS IP address]@[PTD]>
    Max-Forwards: 10
    CSeq: 3 MESSAGE
    User-Agent: EncryptorDiscovery/v1.0
    Call-ID: [value]
    Expires: [seconds]
    Event: presence
    Content-Type: application/pidf+xml
    Content-Length: [value]
Message body
     eXtensible Markup Language
      "<?xml version='1.0' encoding='UTF-8'?>\n"
         "<Plain_Text_Domain>\n"
             "<encryptor>\n"
              "<encryptor_address>AAA.BBB.CCC.DDD</encryptor_address>\n"
              "<encryptor_mask>host or network</encryptor_mask>\n"
              "<encryptor_ttl>in seconds</encryptor_ttl>\n"
              "<database>\n"
                "<address>\n"
                  "<ip>AAA.BBB.CCC.DDD</ip>\n"
                  "<mask>host or network</mask>\n"
                  "<type>host/router/server</type>\n"
                  "<action> insert/delete/enable/disabled</action>\n"
                  "<status> startup/ringing/idle/crashed/active</status>\n"
                "</address>\n"
              "</database>\n"
              "<POC>\n"
                "<name>ISSO</name>\n"
                "<phone>telephone number</phone>\n"
              "</POC>\n"
              "<SA>\n"
                "<key>123456789</key>\n"
                "<mode>tactical/strategic</mode>\n"
              "</SA>\n"
             "</encryptor>\n"
           "<checksum>SHA</checksum>\n"
          "</Plain_Text_Domain>\n"
```

Fig. 1 – MESSAGE message format and content

```
Request-Line: PUBLISH sip:[subscribing PUA IP address]@[PTD] SIP/2.0
    Method: PUBLISH
    [Resent Packet: False]
  Message Header
    Via: SIP/2.0/TCP [PS IP address]:5060;branch=[value]
    From: <sip:[subscribing PUA]@[PTD]>;tag=[value];epid=[value]
    To: <sip:[PS IP address]@[PTD]>
    Max-Forwards: 10
    CSeq: 3 PUBLISH
    User-Agent: EncryptorDiscovery/v1.0
    Call-ID: [value]
    Expires: [seconds]
    Event: presence
    Content-Type: application/pidf+xml
    Content-Length: [value]
  Message body
     eXtensible Markup Language
      "<?xml version='1.0' encoding='UTF-8'?>\n"
        "<Plain_Text_Domain>\n"
           "<encryptor>\n"
            "<encryptor_address>AAA.BBB.CCC.DDD</encryptor_address>\n"
            "<encryptor_mask>host or network</encryptor_mask>\n"
            "<encryptor_ttl>in seconds</encryptor_ttl>\n"
            "<database>\n"
             "<address>\n"
              "<ip>AAA.BBB.CCC.DDD</ip>\n"
              "<mask>host or network</mask>\n"
              "<type>host/router/server</type>\n"
              "<action> insert/delete/enable/disabled</action>\n"
              "<status> startup/ringing/idle/crashed/active</status>\n"
             "</address>\n"
            "</database>\n"
            "<POC>\n"
             "<name>ISSO</name>\n"
             "<phone>telephone number</phone>\n"
            "</POC>\n"
            "<SA>\n"
             "<key>123456789</key>\n"
             "<mode>tactical/strategic</mode>\n"
            "</SA>\n"
           "</encryptor>\n"
          "<checksum>SHA</checksum>\n"
        "</Plain_Text_Domain>\n"
```

Fig. 2 – PUBLISH message format and content

```
Request-Line: SUBSCRIBE sip:[target end system IP address]@[PTD] SIP/2.0
        Method: SUBSCRIBE
        [Resent Packet: False]
    Message Header
        Via: SIP/2.0/TCP [PS IP address]:5060;branch=[value]
        From: <sip:[subscribing PUA IP address]@[PTD]>;tag=[value];epid=[value]
        To: <sip:[PS]@[PTD]>
        Max-Forwards: 10
        CSeq: 5 SUBSCRIBE
        User-Agent: EncryptorDiscovery/v1.0
        Call-ID: [value]
        Expires: 1200
        Accept: application/pidf+xml, application/xpidf+xml
        Event: presence
        Contact: <sip:[subscribing PUA IP address]@[PTD]:5060;transport=tcp>;methods="MESSAGE,
SUBSCRIBE, NOTIFY"
        Content-Length: 0
    Message body
            eXtensible Markup Language
           "<?xml version='1.0' encoding='UTF-8'?>\n"
              "<Plain_Text_Domain>\n"
                 "<encryptor>\n"
                   "<encryptor_address>AAA.BBB.CCC.DDD</encryptor_address>\n"
                   "<encryptor_mask>host or network</encryptor_mask>\n"
                   "<encryptor_ttl>in seconds</encryptor_ttl>\n"
                   "<database>\n"
                     "<address>\n"
                       "<ip>AAA.BBB.CCC.DDD</ip>\n"
                       "<mask>host or network</mask>\n"
                       "<type>host/router/server</type>\n"
                       "<action> insert/delete/enable/disabled</action>\n"
                       "<status> startup/ringing/idle/crashed/active</status>\n"
                     "</address>\n"
                   "</database>\n"
                   "<POC>\n"
                     "<name>ISSO</name>\n"
                     "<phone>telephone number</phone>\n"
                   "</POC>\n"
                   "<SA>\n"
                     "<key>123456789</key>\n"
                     "<mode>tactical/strategic</mode>\n"
                   "</SA>\n"
                 "</encryptor>\n"
               "<checksum>SHA</checksum>\n"
              "</Plain_Text_Domain>\n"
```

Fig. 3 – SUBSCRIBE message format and content

13.4 NOTIFY PTD Database Exchange

Figure 4 details the content of a typical PTD PIDF document. Table 1 defines the PTD PIDF document XML field names. This information may be contained in any or all of the PUBLISH, SUBCRIBE, or NOTIFY messages. Specifically, the first PUBLISH message sent by the encryptor will contain the encryptor’s PTD database. The PS will use the NOTIFY message with an attached PEP PIDF

document to inform both a subscribing and subscribed PUA about the PTD database information pertaining to the subscription request. The NOTIFY “Request-Line” field contains the target end system’s IP address and is used to match an end system IP to a PS PTD encryptor database element. The “To” field contains either the subscribing or subscribed encryptor IP address. The “From” and “Contact” fields will contain the PS IP address.

```
Request-Line: NOTIFY sip:[target end system IP address]@[PTD] SIP/2.0
    Method: NOTIFY
    [Resent Packet: False]
  Message Header
    Via: SIP/2.0/TCP [PS IP address]:5060;branch=[value]
    From: <sip:[PS IP address]@[PTD]>;tag=[value];epid=[value]
    To: <sip:[subscribing or subscribed PUA IP address]@[PTD]>
    Max-Forwards: 10
    CSeq: 5 NOTIFY
    User-Agent: EncryptorDiscovery/v1.0
    Call-ID: [value]
    Expires: 1200
    Accept: application/pidf+xml, application/xpidf+xml
    Event: presence
    Contact: <sip:[PS IP address]@[PTD]:5060;transport=tcp>;methods="MESSAGE, SUBSCRIBE, NOTIFY"
    Content-Length: 0
  Message body
      eXtensible Markup Language
      "<?xml version='1.0' encoding='UTF-8'?>\n"
        "<Plain_Text_Domain>\n"
           "<encryptor>\n"
            "<encryptor_address>AAA.BBB.CCC.DDD</encryptor_address>\n"
            "<encryptor_mask>host or network</encryptor_mask>\n"
            "<encryptor_ttl>in seconds</encryptor_ttl>\n"
            "<database>\n"
             "<address>\n"
              "<ip>AAA.BBB.CCC.DDD</ip>\n"
              "<mask>host or network</mask>\n"
              "<type>host/router/server</type>\n"
              "<action> insert/delete/enable/disabled</action>\n"
              "<status> startup/ringing/idle/crashed/active</status>\n"
             "</address>\n"
            "</database>\n"
            "<POC>\n"
             "<name>ISSO</name>\n"
             "<phone>telephone number</phone>\n"
            "</POC>\n"
            "<SA>\n"
             "<key>123456789</key>\n"
             "<mode>tactical/strategic</mode>\n"
            "</SA>\n"
           "</encryptor>\n"
          "<checksum>SHA</checksum>\n"
        "</Plain_Text_Domain>\n"
```

Fig. 4 – NOTIFY PTD PIDF XML fields

Table 1 – PTD PIDF XML Field Names

Field Entry	Required/Optional	Definition
Plain_Text_Domain	Required	PTD common name
Encryptor	Required	Encryptor IP address
Ttl	Required	Time-to-live in seconds
Database	Required	PTD database section
Address	Required	PTD device IP address
Mask	Required	PTD device IP mask
Type	Required	PTD device type
Action	Optional	Default: enable
Status	Optional	Device state information
POC	Optional	ISSO Point-of-contact section
Name	Optional	ISSO name
phone	Optional	ISSO Phone number
SA	Optional	SA section
Key	Optional	SA public key
Mode	Optional	Tactical = hop-by-hop or Strategic = host-by-host
Checksum	Required	Secure Hash Algorithm

14. SIP OVERCOMES SINGLE POINTS OF FAILURE

Through the use of the SIP proxy server, single points of failure can be avoided because the UAS can forward requests to other UASs until it hits its destination. If it is a re-direct server, then it returns the direct address of a target encryption device UAC. Another method to overcome single points of failure, and the preferred method used by this design specification, is the SIP P2P implementation using Distributed Hash Table (DHT) architecture. (Note: the initial proof-of-concept prototype will not include a P2P configuration.)

15. SIP ENCRYPTION DEVICE PTD DISCOVERY SERVICE

Typical encryption device specifications state PTD discovery services. For example, the following is an extraction from an encryption device requirements specification:

a. The Generic Discovery Client Extension will describe the requirements on an encryptor to support Peer Discovery independently of the server specific approach. The protocols used by the Generic Discovery Client allow the encryptor to request and provide information about the reachability of PTD networks. The functionalities should work regardless of the network architecture and independent of whether a Multicast or Client/Server discovery mechanism is being used.

All the benefits associated with the SIP security architecture can be exploited by the SIP Encryption Device PTD Discovery Service. As just one example, DoS attacks can be avoided through coupling of SIP P2P with peering UASs and further exploiting protected channels such as IPSec or Secure Socket Layer Virtual Private Network (SSL VPN) services. SIP Certificate Authentication can provide first level attack protection. Further, existing IP security (IPSec) and SSL VPN architectures can easily be integrated into this SIP Encryption Device PTD Discovery design to provide additional security.

16. SIP ENCRYPTOR PTD DISCOVERY CONCEPT OF OPERATION (CONOP)

The following encryption device specification extractions are used to scope the CONOP for this design.

a. In a Client/Server architecture, the encryption device would be configured with a list of Plain Text (PT) and Cypher Text (CT) addresses of Servers to which it will register its prefixes. The encryption device would send a Registration Message containing a list of the Prefixes the encryptor has in its Local Enclave Prefix Table to all of the servers in the list. This Registration will also contain the lifetime that the encryptor anticipates the prefixes will be valid. If any changes occur to the Local Enclave Prefix Table (e.g., removing or adding a prefix) a new Registration Message would be sent. The encryptor would send Registration messages (to the Servers it is configured with) until receiving an Acknowledgment message.
b. The encryption device will also be configured with the PT and CT address of the Server(s) to which it will send Queries. (Note: this will often be the server it sends Registration information to, but it does not necessarily have to be the same.) These Queries contain the PT addresses or prefixes that the encryptor is attempting to reach.

When an encryption device is first activated within a network, like other SIP aware devices, it acts as a SIP User Agent Client (UAC) programmed to begin establishing an association with a SIP User Agent Server (UAS). In other words, the encryptor will “register” its URI with the network UAS. In so doing, it will inform other encryptors of its presence through the UAS. Querying encryptors will discover other encryption devices through the use of the SIP INVITE SDP message or PEP PIDF messages to either the UAS or UAC in the case of a P2P association.

17. ENCRYPTION DEVICE REGISTRATION REGISTER CONOPS

The encryptor uses its IP as its current location and registers the IP with the server in the form **name@IP**. This information is stored by the server (or in the case of P2P the target peer) and later recalled when another encryptor needs to establish a SA with this registering encryptor. This REGISTER information is also used to initialize the Plain Text Domain reachability database, which will be populated by either the sourcing encryptor’s INVITE SDP information or REGISTER Path field information. Furthermore, this REGISTER dialogue will help with mobile encryption devices. Say there is to be a tactical deployment of an encryptor. After the encryptor is deployed and first turned on, it will register with a preselected UAS using its tactical IP. The INVITE “Expire” field reflects the duration for which this registration will be valid. So, as long as the encryptor is deployed at its current location, it will refresh its registration from time to time. When the encryptor moves from its current location to a new IP domain, it will re-register with the UAS, thus updating its location information. Figure 5 illustrates this registration process.

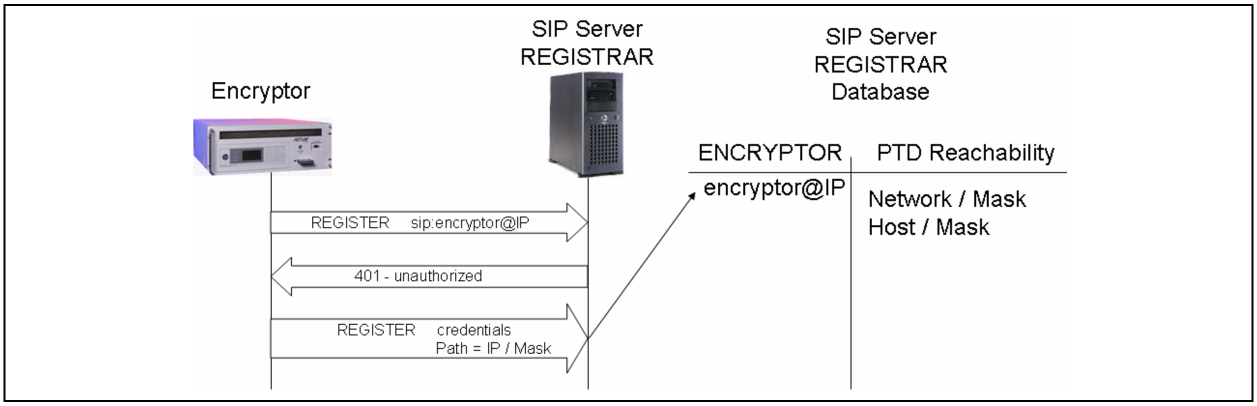

Fig. 5 – SIP registration

18. PLAIN TEXT DOMAIN UAC TO UAS INVITE WITH SDP CONOP

After an encryptor (the UAC) registers with the UAS, it will initiate an INVITE message dialogue with the UAS (or P2P UAC) containing PTD information data. The UAC will construct the INVITE SDP message with its PTD Reachability information. The UAS will extract the PTD information from the INVITE SDP message, inserting or update the data into the UAS PTD Reachability database. As with typical SIP INVITE SDP dialogue sequences, the UAS (or P2P UAC) will respond to the initial UAC INVITE with a 180 - Ringing message to indicate the received message is in process. Once the PTD information has been successfully updated within the UAS PTD database, the UAS will send a 200 - OK response message, thus informing the UAC that it is ready to forward SA information associated to the UAC. Figure 6 shows the message exchange process.

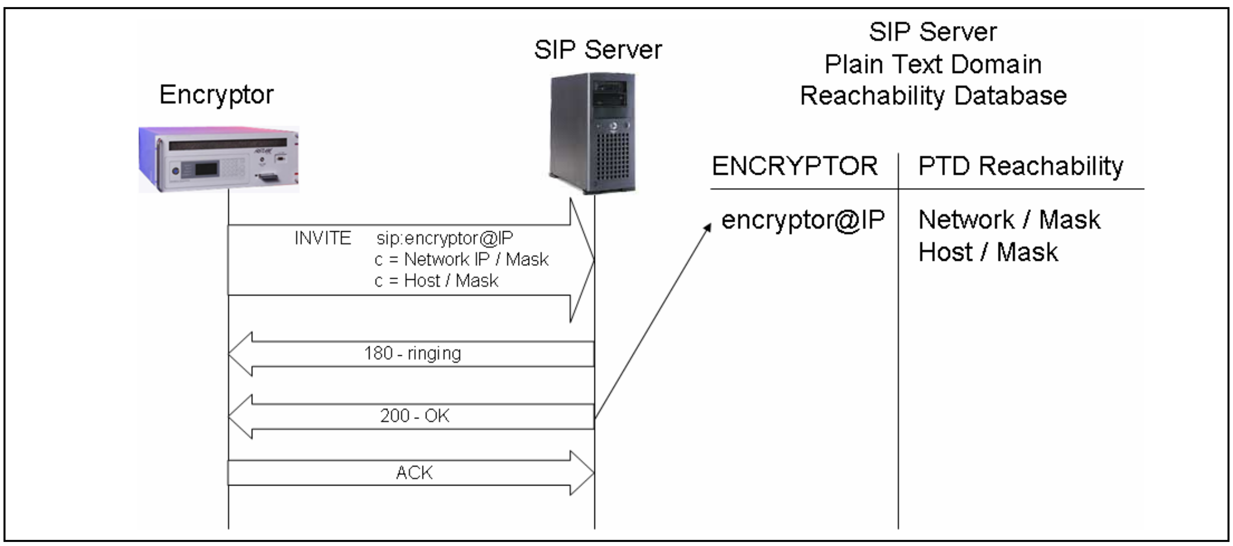

Fig. 6 – Encryptor INVITE UAC to UAS SDP dialogue

19. PLAIN TEXT DOMAIN UAS TO UAC INVITE WITH SDP CONOP

When a front ending encryptor receives a request from the encryptor's PTD client for a network or host SA, it will generate a SIP INVITE message to the UAS using the network or host IP of the PTD client as the SIP "To" information. Using the SIP "To" name@IP information, the UAS will query its database, and if a matching PTD network or host is found, it will build a INVITE SDP message with a single "c=[network/mask or host/mask]" entry with the target encryptor's PTD information found in the UAS PTD database. Then the UAS will query the PTD database using the requester's INVITE "Contact" field information for a matching PTD Reachability entry for the sourcing encryptor. The UAS then builds an INVITE SDP message to be sent to the target encryptor, populating a single SDP "c=[network/mask or host/mask]" entry with the sourcing encryptor's PTD Reachability information. Prior to querying its data base, the UAS will respond to the requesting encryptor with a 180 - Ringing SIP message. Then after the two INVITE messages, one for the requesting encryptor and one for the target encryptor, are successfully generated, the UAS forwards these INVITE SDP messages to each encryptor as illustrated by Fig. 7.

If at any time the UAS cannot successfully generate and forward these INVITE SDP messages, the UAS will forward an SDP 5XX server failure message to the requesting encryptor terminating the SA sequence.

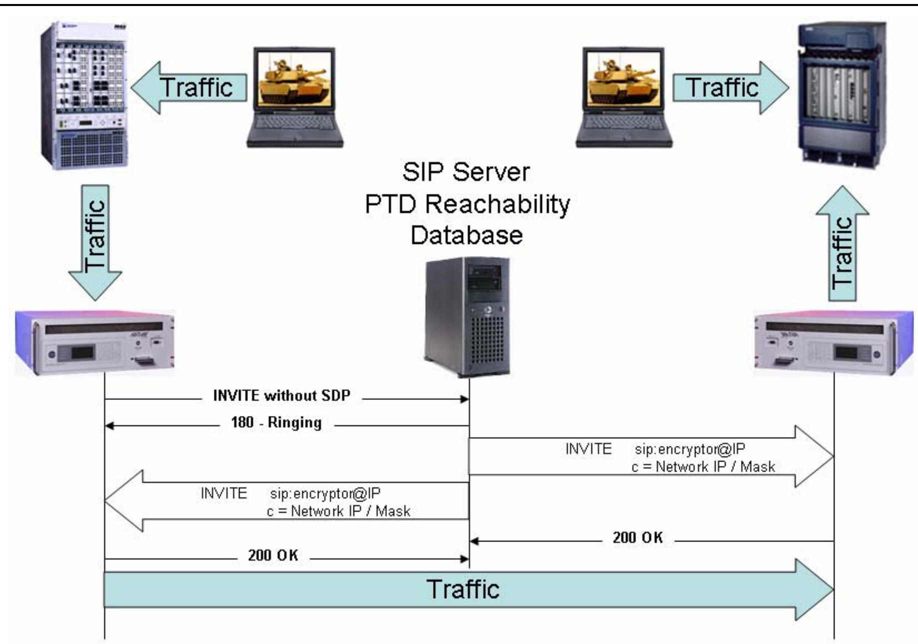

Fig. 7 – UAS INVITE SDP message traffic

20. PLAIN TEXT DOMAIN UAC RECEIVED INVITE WITH SINGLE SDP ENTRY CONOP

When an encryptor receives an UAS INVITE message with a single SDP entry and it is not in a P2P relationship with the sourcing UAS, it will use the SDP "c=[network/mask or host/mask]" field to establish a SA with the encryptor whose location is specified in the INVITE SDP "Contact" field. If the encryptor is a P2P peer to the sourcing UAS, then this information will be used to update the local PTD Reachability database and used as SDP information to establish a SA with the peer.

21. PLAIN TEXT DOMAIN PEP PIDF (PUBLISH, SUBSCRIBE, AND NOTIFY) CONOP

The PEP PIDF message is considered the preferred method for exchanging PTD database information. This method begins after the encryptor has registered with the UAS (PS). Acting as a PUA, the encryptor will construct a PUBLISH message containing a PEP PIDF document populated with encryptor's PTD database and forward the PUBLISH PEP PIDF document to the PS (see Fig. 8).

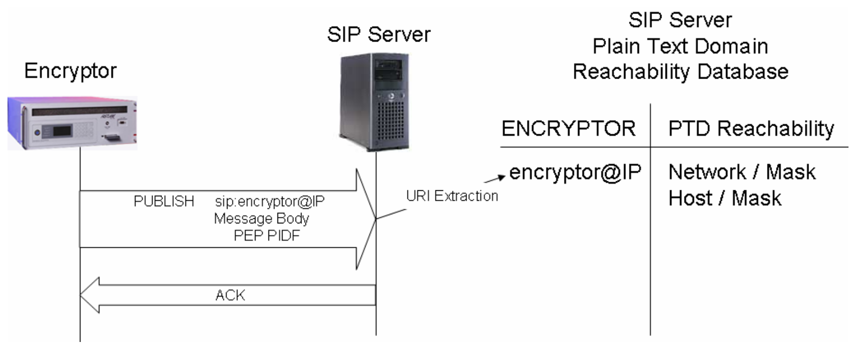

Fig. 8 – PUBLISH PEP PIDF message PTD database update traffic

This PIDF document contains PTD information in the format previously defined. The PIDF also includes a time-to-live (ttl) field that is used to assure that the PTD information is current. The ttl field is GMT stamped at the time the PEP PIDF message is first forwarded to the PS (or peering P2P UAC). The ttl field is used as part of a PTD heartbeat message. This heartbeat message may or may not coincide with normal SIP SUBSCRIBE expiration subscription messages. The ttl expiration value is set according to ISSO policy, therefore, the ttl is an arbitrary value based on the PTD security policy. After the encryptor is acknowledged for the PUBLISH PEP PIDF message, it can begin issuing SUBSCRIBE messages for any target end system. The SUBSCRIBE message is used by the subscribing encyrptor to request from the PS the IP address of the encryptor that is protecting the target end system. In response to the subscription message, the PS will issue NOTIFY messages to the subscribing encryptor containing PTD database information pertaining to the target end system. The PS will learn the target end system's fronting encryptor IP address by comparing the end system IP address extracted from the SUBSCRIBE "Request-Line" with the PS PTD database. The PS extracts from the PTD database the associated PTD information that matches the end system's IP address and inserts this information into a NOTIFY message's PEP PIDF document. The PS then forwards the message to the subscribing encyrptor. The subscribing encryptor learns the target end system's encryptor IP address by searching for "PTD_IP_address" entries in the received NOTIFY PIDF document. Once a match is found, it extracts from the PIDF document the corresponding encyptor IP address. It then uses this IP address to establish the security association between PTDs. The PS will also forward the same NOTIFY message to the subscribed encryptor. The subscribed encryptor will also look for and extract the subscribing encryptor's IP address. When a match is found, it will establish a security association with the subscribing encryptor. At this point, protected traffic between PTD end systems can be conducted. This process is illustrated in Fig. 9.

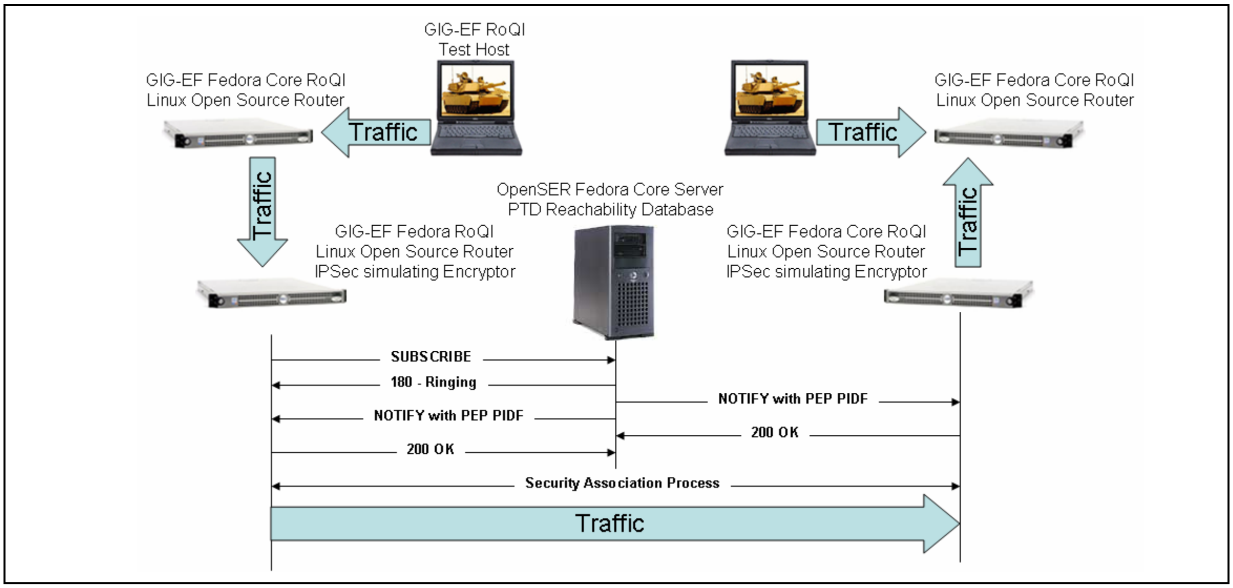

Fig. 9 – PUBLISH PEP PIDF message traffic flow

22. SIP MESSAGE PLAIN TEXT DOMAIN PEP PIDF CONOP

The SIP MESSAGE PEP PIDF message is considered the preferred method for low bandwidth PTD database information traffic exchanges. This method does not require an encryptor to register with the UAS (PS). Used for tactical traffic, MESSAGE traffic is secure using other methods such as encrypted Virtual Private Network overlays, typically using IPSec “tactical” tunnels. Acting as a PUA, the encryptor will construct a MESSAGE packet containing a PEP PIDF document populated with the encryptor’s PTD database and forward the MESSAGE PEP PIDF document to a peering PUA (see Fig. 10).

23. PLAIN TEXT DOMAIN REACHABILITY TRAFFIC FLOW CONOP

Traffic flow from a PTD client to another PTD client consists of the sourcing client traffic first triggering an SA between PTD fronting encryptors as detailed in the paragraphs above that outline the INVITE CONOP message traffic flow. Once the fronting encryptors have established peering Security Associations, the PTD clients can begin exchanging traffic as illustrated in Fig. 11.

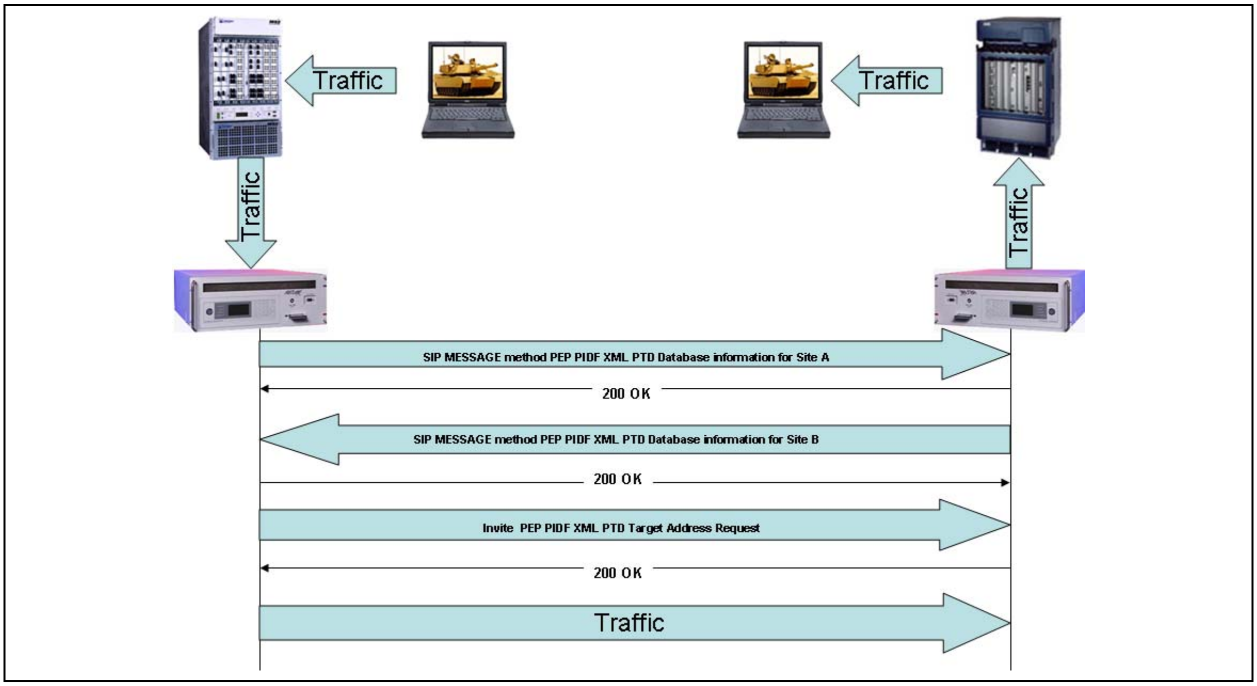

Fig. 10 – MESSAGE PEP PIDF message traffic flow

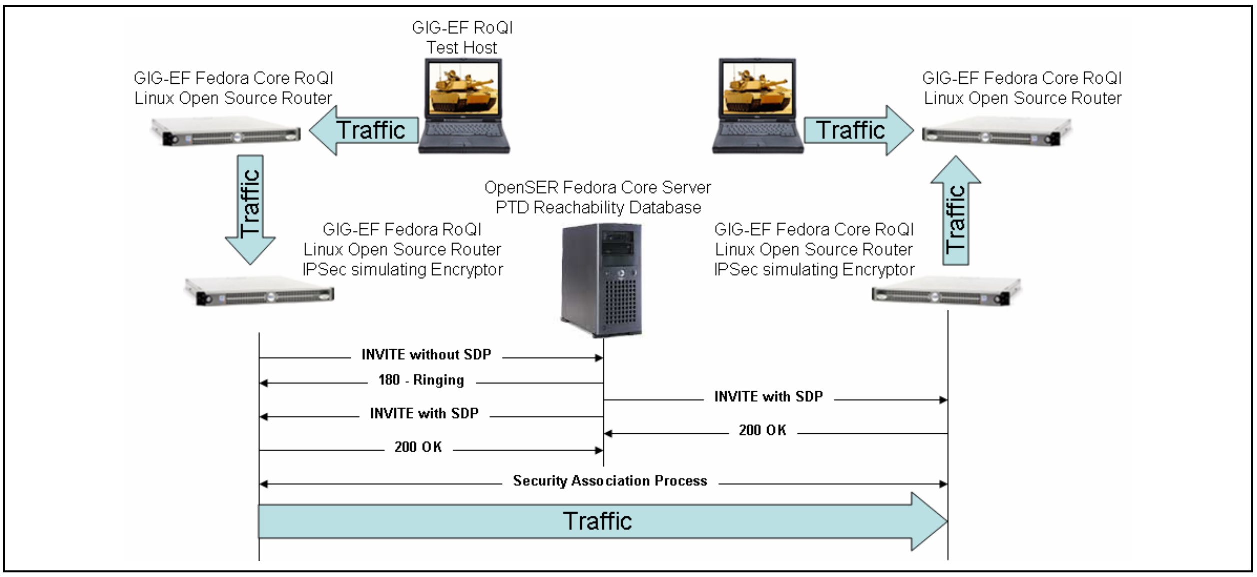

Fig. 11 – UAC traffic flow sequence

24. PROTOTYPE ARCHITECTURE OF THIS SIP PTD DISCOVERY SERVICE

To validate this architecture, a prototype demonstration configuration will be implemented. This prototype will use existing Global Information Grid Evaluation Facilities (GIG-EF) Routing over Quality of Service Infrastructure (RoQI) Linux Open Source Routers (RoQILOSR) as network components and RoQI test hosts as PTD clients. To simulate SIP PTD Discovery Service encryption devices, an additional RoQILOSR, called the Open Source Linux Simulated Encryptor (OSLSE), will be inserted

between the RoQILOSR Customer-Edge (CE) router and the RoQILOSR Provider-Edge (PE) router, as illustrated in Fig. 12. An OpenSER (Open SIP Express Router running a Fedora Core 6 Operating System) server will be implemented as the SIP Server PTD Reachability Database. With these components, it is envisioned the prototype architecture will successfully exchange PTD control and data traffic. To simulate the encrypted SA path, the OSLSE will exploit IPSec network-to-network protocol to simulate protected PTD enclave interconnected paths to prove the feasibility of the SIP PTD Discovery Service.

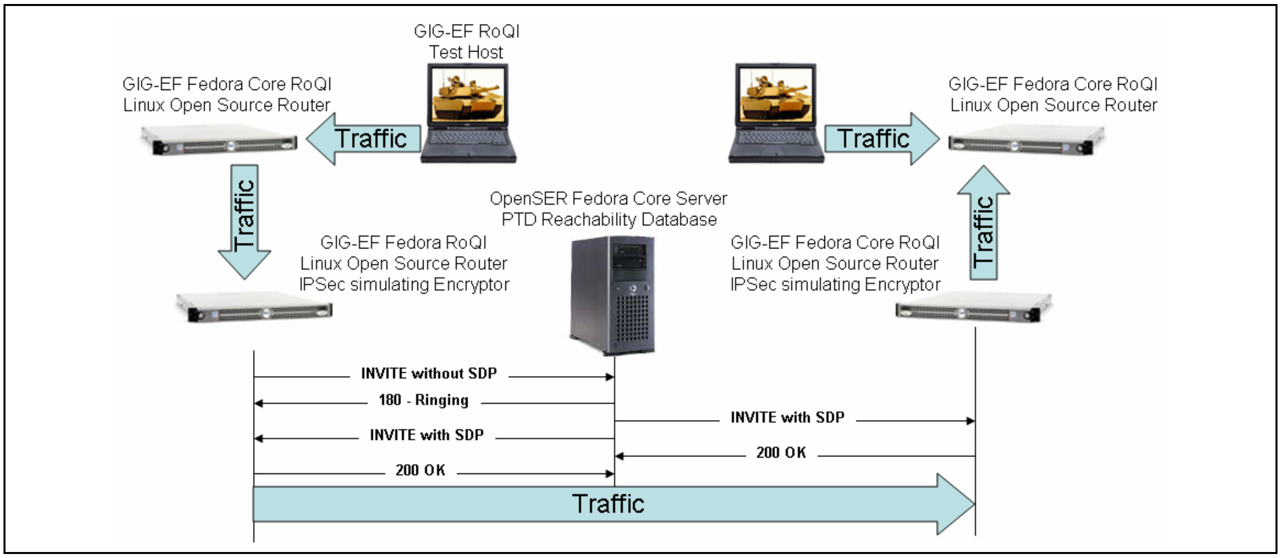

Fig. 12 – SIP PTD Discovery Service GIG-EF RoQILOSR prototype architecture

25. SIP PTD DISCOVERY SERVICE PROTOTYPE ARCHITECTURE SIMULATED ENCRYPTOR

The system architecture components of the simulated encryptor just discussed include Fedora Core 6 (FC6) as the Operating System, QUAGGA as the IP routing stack, and the FC6 implementation of IPSec as the encryption engine. To prototype the SIP UAC component, the Sofia-SIP[2] stack will be exploited. Figure 13 illustrates the various components and how these components relate.

[2] Sofia-SIP is an open-source SIP User-Agent library, compliant with the IETF RFC3261 specification.

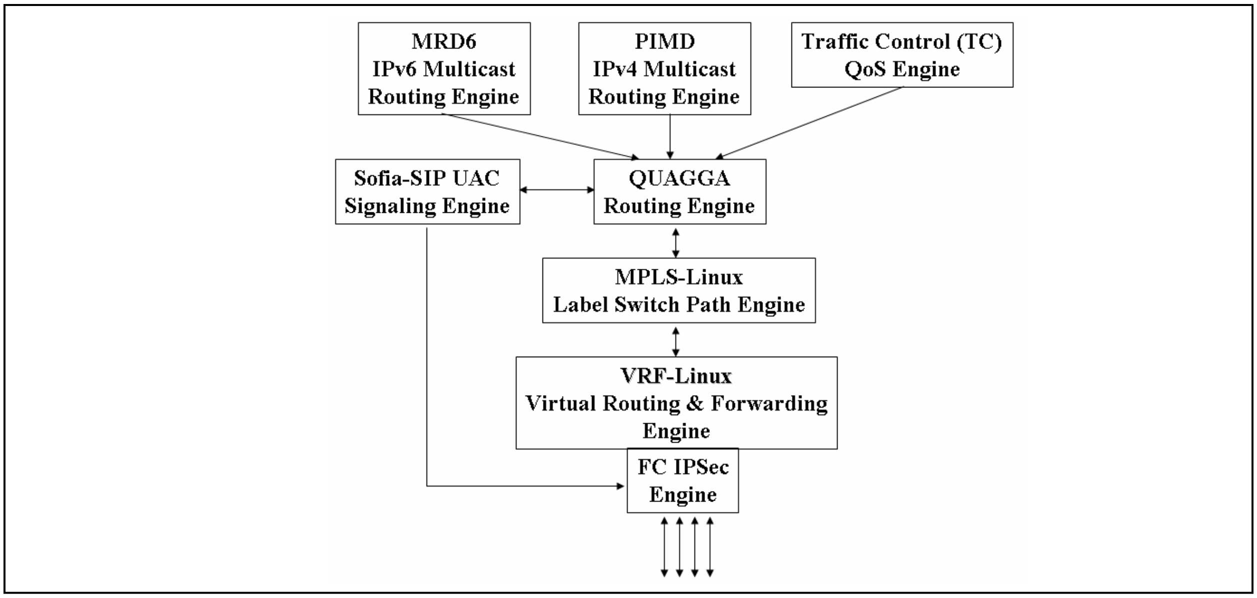

Fig. 13 – Prototype SIP PTD architecture simulated encryption device

26. SIP PTD DISCOVERY SERVICE PROTOTYPE ARCHITECTURE OPENSER CONFIGURATION

The architecture components of the OpenSER PTD Database just discussed include Fedora Core 6 as the OS, QUAGGA as the IP routing stack, and Open SIP Express Router (OpenSER). To prototype the SIP UAS database component, a modification of the OpenSER MySQL database will be exploited. Figure 14 illustrates the various components and how these components relate.

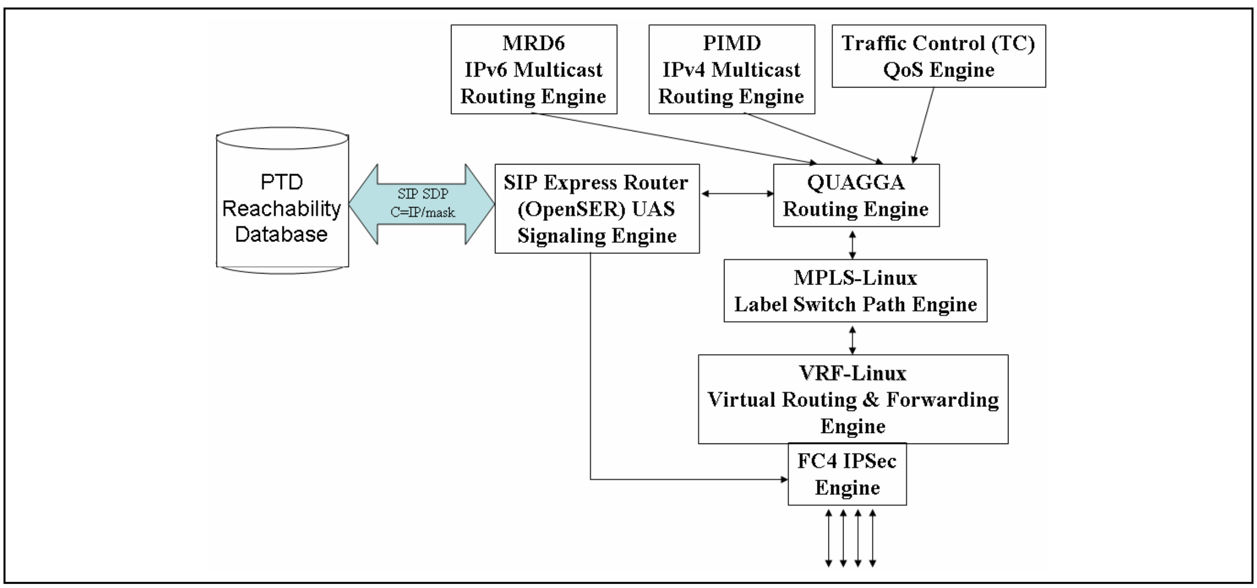

Fig. 14 – Prototype SIP PTD reachability database architecture

27. IPSEC PROTECTED CONTROL CHANNEL

To demonstrate a possible solution for protecting the SIP Control Plane control channels, an out-of-band IPSec control channel will be deployed within this prototype. It should be noted that out-of-band within this specification does not necessarily mean a completely separate physical communication medium. For example, the protected channel may be either a separate Virtual Local Area Network (VLAN) or MultiProtocol Label Switching (MPLS) Label Switched Path. For this specification, out-of-band will refer to a channel that does not carry user data traffic. This control channel will be implemented as a host-to-host single hop channel. Specifically, it will be a configured channel between peering network components. For example, two peering layer three routers with adjacent interfaces will communicate over their adjacent interfaces through an IPSec protected channel. Figure 15 illustrates this implemented configuration. Similar to how existing FASTLANE network management is implemented, the SIP Control Plane Discovery Service will be installed behind a protecting encryption device. Therefore, client encryptors will have a statically preconfigured security association to the SIP Control Plane Discovery Service server's protecting encryptor.

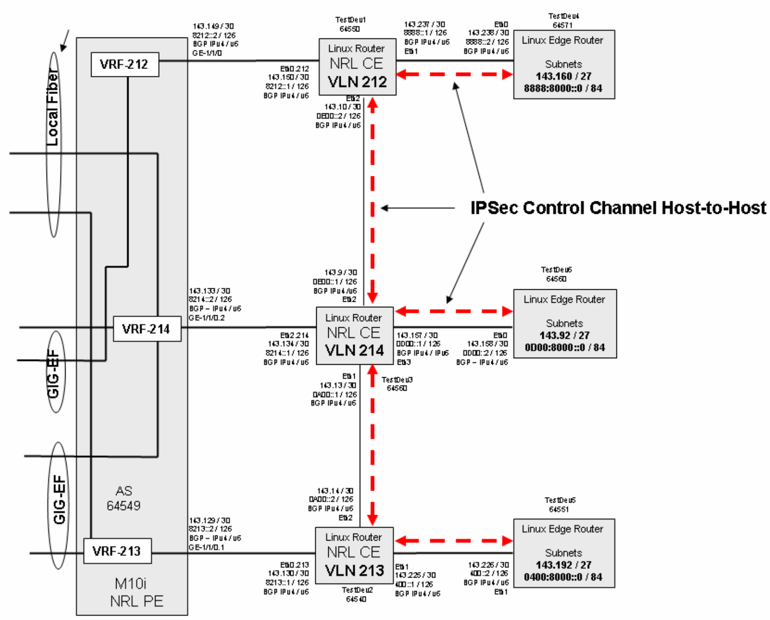

Fig. 15 – IPSec control channel implementation

28. SIP ENCRYPTION DEVICE PTD DISCOVERY SERVICE WORKING PROTOTYPE

To demonstrate the feasibility of this architecture, an initial working prototype was developed consisting of two components, an "Initiator" and a "Responder." The Initiator's function is to create a PTD database MESSAGE packet and forward the message to a peering PUA's Responder. The Responder is responsible for ingesting the incoming SIP message, extracting the PTD PEP PIDF XML data from the MESSAGE message body. The Appendix contains the Initiator and Responder code. Figure 16 is an example of the Responder output of the PTD PEP PIDF XML received from the Initiator.

Incoming SIP Message XML document - From: sip:127.0.0.1:15062
Subject: Plain Text Domain Database Update
Plain Text Domain Database is being updated


```
<?xml version='1.0' encoding='UTF-8'?>
<Plain_Text_Domain>
<encryptor>
<encryptor_address>10.1.1.1</encryptor_address>
<encryptor_mask>255.255.255.0</encryptor_mask>
<encryptor_ttl>120</encryptor_ttl>
<database>
<address>
<ip>192.168.1.0</ip>
<mask>255.255.255.0</mask>
<type>network</type>
<action>insert</action>
<status>active</status>
</address>
<address>
<ip>192.168.2.1</ip>
<mask>255.255.255.225</mask>
<type>host</type>
<action>insert</action>
<status>active</status>
</address>
<address>
<ip>10.1.1.2</ip>
<mask>255.255.255.255</mask>
<type>router</type>
<action>insert</action>
<status>active</status>
</address>
</database>
<POC>
<name>NRL</name>
<phone>111-111-1111</phone>
</POC>
<SA>
<key>123456789</key>
<mode>tactical</mode>
</SA>
</encryptor>
<checksum>SHA-123456789</checksum>
</Plain_Text_Domain>
```


Plain Text Domain Database has been updated

Fig. 16 – PTD PEP PIDF XML Responder output as received from the Initiator

29. CONCLUSION

SIP makes an ideal encryption device PTD discovery solution. Further, using SIP, it becomes a simple implementation effort. No modification to the SIP standard is required to support any encryption

device discovery or to any implementation of the standard. This would further allow IA implementations to remain viable because of SIP's modular design that allows for expansion and enhancements while still supporting existing implementations. Also, by building an integrated UAC to UAS association coupled with a SIP P2P DHT, IPSec, and SSL VPN architecture, it will be feasible to completely overcome any single point of failure secure encryption device discovery process.

The working prototype also demonstrated how the Initiator and Responder can be enhanced to provide a complete IP control plane using the architecture developed in this specification. For example, the Responder is being modified to reside in servers, clients as well as encryptors, to manage various IP traffic situations such as Quality-of-Service negotiations, Network Management Systems (NMS) status information updates, and Advanced Question Answer dialogues, and it is envisioned that key management can exploit this architecture. These are as just a few possible examples of how this architecture will help manage IP traffic, particularly in tactical networks.

REFERENCES

1. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261, pp. 1-269, June 2002.

2. R. Sparks, *The Session Initiation Protocol (SIP) Refer Method*, RFC 3515, pp. 1-23, April 2003.

3. A.B. Roach, *SIP-Specific Event Notification*, RFC 3265, pp. 1-32, June 2002.

BIBLIOGRAPHY

1. M. Handley and V. Jacobson, *SDP: Session Description Protocol*, RFC 2327, pp. 1-42, April 1998.

2. S. Kent and R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, pp. 1-66, November 1998.

3. J. Rosenberg and H. Schulzrinne, *An Offer/Answer Model with the Session Description Protocol (SDP)*, RFC 3264, pp. 1-25, June 2002.

4. B. Campbell and R. Sparks, *Control of Service Context using SIP Request-URI*, RFC 3087, pp. 1-39, April 2001.

5. F. Le Faucheur and W. Lai, *Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering*, RFC 3564, pp. 1-22, July 2003.

6. S. Kent and R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, pp. 1-66, November 1998.

7. J.-L. Le Roux, J.-P. Vasseur, and J. Boyle, *Requirements for Inter-Area MPLS Traffic Engineering*, RFC 4105, pp. 1-22, June 2005.

8. F. Le Faucheur, (ed.), L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen, *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services*, RFC 3270, pp. 1-64, May 2002.

9. M. Lasserre and V. Kompella, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*, RFC 4762, pp. 1-31, January 2007.

10. E.C. Rosen, J. De Clercq, O. Paridaens, and Y. T'Joens, *Architecture for the Use of PE-PE IPsec Tunnels in BGP/MPLS IP VPNs*, draft-ietf-l3vpn-ipsec-2547-05.txt, pp. 1-18, August 2005.

11. D. Willis and B. Hoeneisen, *Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts*, RFC 3327, pp. 1-17, December 2002.

12. H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson, *Presence Information Data Format (PIDF)*, RFC 3863, pp. 1-28, August 2004.

13. J. Rosenberg, *A Presence Event Package for the Session Initiation Protocol (SIP)*, RFC 3856, pp. 1-27, August 2004.

14. J. Peterson, *Common Profile for Presence (CPP)*, RFC 3859, pp. 1-15, August 2004.

15. T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau, *Extensible Markup Language (XML) 1.0* (Fourth Edition), sections 1 through I, September 2006.

16. S. Kent, *IP Encapsulating Security Payload (ESP)*, RFC 4303, pp. 1-44, December 2005.

17. W. Almesberger, *Linux Traffic Control – Next Generation*, 18 Oct 2002.

18. Sofia-SIP Sourceforge: http://sofia-sip.sourceforge.net.

19. S. Friedl Unixwiz.net Tech Tips: http://www.unixwiz.net/techtips/iguide-ipsec.html, August, 2005.

20. General Dynamics: http://www.gdc4s.com.

Appendix
PROTOTYPE CODE

A-1. INITIATOR

```
/*
 *
 * This application will send a MESSAGE with a PEP PIDF XML document
 * from <sip:localhost:5062> to <sip:localhost:5060>
 */

#include <stdio.h>
#include <sofia-sip/nua.h>

char Plain_Text_Domain[] = {
        "<?xml version='1.0' encoding='UTF-8'?>\n"
          "<Plain_Text_Domain>\n"
              "<encryptor>\n"
                "<encryptor_address>10.1.1.1</encryptor_address>\n"
                "<encryptor_mask>255.255.255.0</encryptor_mask>\n"
                "<encryptor_ttl>120</encryptor_ttl>\n"
                "<database>\n"
                  "<address>\n"
                    "<ip>192.168.1.0</ip>\n"
                    "<mask>255.255.255.0</mask>\n"
                    "<type>network</type>\n"
                    "<action>insert</action>\n"
                    "<status>active</status>\n"
                  "</address>\n"
                  "<address>\n"
                    "<ip>192.168.2.1</ip>\n"
                    "<mask>255.255.255.225</mask>\n"
                    "<type>host</type>\n"
                    "<action>insert</action>\n"
                    "<status>active</status>\n"
                  "</address>\n"
                  "<address>\n"
                    "<ip>10.1.1.2</ip>\n"
                    "<mask>255.255.255.255</mask>\n"
                    "<type>router</type>\n"
                    "<action>insert</action>\n"
                    "<status>active</status>\n"
                  "</address>\n"
                "</database>\n"
                "<POC>\n"
                  "<name>NRL</name>\n"
                  "<phone>111-111-1111</phone>\n"
                "</POC>\n"
                "<SA>\n"
                  "<key>123456789</key>\n"
                  "<mode>tactical</mode>\n"
                "</SA>\n"
              "</encryptor>\n"
            "<checksum>SHA-123456789</checksum>\n"
          "</Plain_Text_Domain>\n\0\0"};

void send_message (nua_t *nua);

void event_callback(nua_event_t   event,
                    int           status,
                    char const   *phrase,
                    nua_t        *nua,
                    nua_magic_t  *magic,
                    nua_handle_t *nh,
                    nua_hmagic_t *hmagic,
                    sip_t const  *sip,
                    tagi_t        tags[]);

int
main (int argc, char *argv[])
{
  su_root_t *root;
  nua_t *nua;

  /* Initialize Sofia-SIP library and create event loop */

  su_init ();
  root = su_root_create (NULL);

  /* Create a user agent instance. Caller and callee should bind to different
   * address to avoid conflicts. The stack will call the 'event_callback()'
   * callback when events such as succesful registration to network,
   * an incoming call, etc, occur.
   */
  nua = nua_create(root, /* Event loop */
                 event_callback, /* Callback for processing events */
                 NULL, /* Additional data to pass to callback */
                                   NUTAG_URL("sip:127.0.0.1:15062"), /* Address to
bind to */
                 TAG_END()); /* Last tag should always finish the sequence */

  send_message (nua);

  /* Run event loop */
  su_root_run (root);

  /* Destroy allocated resources */
  nua_destroy (nua);
  su_root_destroy (root);
  su_deinit ();

  return 0;
}

/* This callback will be called by SIP stack to
 * process incoming events
 */
void event_callback(nua_event_t   event,
                    int           status,
                    char const   *phrase,
                    nua_t        *nua,
                    nua_magic_t  *magic,
                    nua_handle_t *nh,
                    nua_hmagic_t *hmagic,
                    sip_t const  *sip,
                    tagi_t        tags[])
{
  printf("Incoming SIP Message XML acknowledgement - From: %s%s"
URL_PRINT_FORMAT "\n",
        sip->sip_from->a_display ? sip->sip_from->a_display : "",
        sip->sip_from->a_display ? " " : "",
        URL_PRINT_ARGS(sip->sip_from->a_url));

  if (sip->sip_subject) {
    printf("Subject: %s\n", sip->sip_subject->g_value);
  }
  if (sip->sip_payload) {
    fwrite(sip->sip_payload->pl_data, sip->sip_payload->pl_len, 1, stdout);
    fputs("\n", stdout);
  }
}

/* Create a communication handle, send MESSAGE PEP PIDF Plain Text Domain
XML File with it and destroy it */
void send_message (nua_t *nua)
{
  nua_handle_t *handle;
  int PTD_len = 0;

  handle = nua_handle(nua,
                   NULL,
                   SIPTAG_TO_STR("sip:127.0.0.1:15060"),
                   TAG_END());

  PTD_len = sizeof(Plain_Text_Domain) - 1;

  nua_message(handle,
           SIPTAG_SUBJECT_STR("Plain Text Domain Database Update"),
                       SIPTAG_CONTENT_TYPE_STR("text/plain"),
                       SIPTAG_PAYLOAD_STR(Plain_Text_Domain),
                       TAG_END());

  nua_handle_destroy (handle);
}
```

A-2. RESPONDER

```
/*
 *
 * This application will receive a MESSAGE PEP PIDF XML document
 * from <sip:localhost:5062> to <sip:localhost:5060>
 */

#include <stdio.h>
#include <sofia-sip/nua.h>

void event_callback(nua_event_t   event,
                    int           status,
                    char const   *phrase,
                    nua_t        *nua,
                    nua_magic_t  *magic,
                    nua_handle_t *nh,
                    nua_hmagic_t *hmagic,
                    sip_t const  *sip,
                    tagi_t        tags[]);

void send_message (nua_t *nua);
int UpdatePTDMySQLDatabase (sip_t const  *sip);
int PTD_status;

int
main (int argc, char *argv[])
{
  su_root_t *root;
  nua_t *nua;

  /* Initialize Sofia-SIP library and create event loop */

  su_init ();
  root = su_root_create (NULL);

  /* Create a user agent instance. Caller and callee should bind to different
   * address to avoid conflicts. The stack will call the 'event_callback()'
   * callback when events such as succesful registration to network,
   * an incoming call, etc, occur.
   */
  nua = nua_create(root, /* Event loop */
                   event_callback, /* Callback for processing events */
                   NULL, /* Additional data to pass to callback */
                                  NUTAG_URL("sip:127.0.0.1:15060"),
                   TAG_END()); /* Last tag should always finish the sequence */

  /* Run event loop */
  su_root_run (root);

  /* Destroy allocated resources */
  nua_destroy (nua);
  su_root_destroy (root);
  su_deinit ();

  return 0;
}

/* This callback will be called by SIP stack to
 * process incoming events
 */
void event_callback(nua_event_t   event,
                    int           status,
                    char const   *phrase,
                    nua_t        *nua,
                    nua_magic_t  *magic,
                    nua_handle_t *nh,
                    nua_hmagic_t *hmagic,
                    sip_t const  *sip,
                    tagi_t        tags[])
{
  printf("Incoming SIP Message XML document - From: %s%s"
URL_PRINT_FORMAT "\n",
         sip->sip_from->a_display ? sip->sip_from->a_display : "",
         sip->sip_from->a_display ? " " : "",
         URL_PRINT_ARGS(sip->sip_from->a_url));

  if (sip->sip_subject) {
    printf("Subject: %s\n", sip->sip_subject->g_value);
  }

//  if (sip->sip_payload) {
//    fwrite(sip->sip_payload->pl_data, sip->sip_payload->pl_len, 1, stdout);
//    fputs("\n", stdout);
//  }
```

```
  /*------------------------------------------------
  * Extract the PTD information and update the
  * Plain Text Domain Database with the new IP info
  * which associates an Encryptor to red side
  * systems, network devices, etc.
  *-----------------------------------------------*/
  PTD_status = UpdatePTDMySQLDatabase (sip);

  /*-------------------------------------------------
  * Tell device PTD PEP PIDF database update received
  * and the local PTD MySQL databsse has been updated
  *------------------------------------------------*/
  send_message (nua);
}

/* Create a communication handle, send MESSAGE PEP PIDF Plain Text Domain
XML File with it and destroy it */
void send_message (nua_t *nua)
{
  nua_handle_t *handle;
  int PTD_len = 0;

  handle = nua_handle(nua,
                      NULL,
                      SIPTAG_TO_STR("sip:127.0.0.1:15062"),
                      TAG_END());

  nua_message(handle,
              SIPTAG_SUBJECT_STR("Plain Text Domain Database Update Received"),
                      SIPTAG_CONTENT_TYPE_STR("text/plain"),
                      SIPTAG_PAYLOAD_STR("Plain Text Domain Database update
completed\n"),
                      TAG_END());

  nua_handle_destroy (handle);
}

/*------------------------------------------------------
* Update the Plain Text Domain Database
*-----------------------------------------------------*/
int UpdatePTDMySQLDatabase (sip_t const  *sip)
{
  printf("Plain Text Domain Database is being updated\n\n");
  if (sip->sip_payload) {
    fwrite(sip->sip_payload->pl_data, sip->sip_payload->pl_len, 1, stdout);
    fputs("\n", stdout);
  }
  printf("Plain Text Domain Database has been updated\n\n");
}
```